

## DOCUMENT RESUME

ED 304 107

IR 013 666

AUTHOR Adamson, Eadie  
TITLE LogoWriter Games: Tips and Tools.  
PUB DATE 88  
NOTE 6p.; Paper presented at the Great Lakes/East Coast LOGO Conference (Cleveland, OH, May 6-7, 1988).  
PUB TYPE Guides - Classroom Use - Guides (For Teachers) (052)  
-- Computer Programs (101) -- Speeches/Conference Papers (150)  
  
EDRS PRICE MF01/PC01 Plus Postage.  
DESCRIPTORS \*Class Activities; \*Computer Games; Computer Software; Microcomputers; \*Problem Solving;  
IDENTIFIERS \*Programing; Secondary Education; Teaching Methods  
\*LOGO Programing Language

## ABSTRACT

This paper provides guidelines and programming commands for using LogoWriter activity cards in game-writing as a way to develop student programming skills. Suggestions are made for the specific activity cards to use, and solutions are provided for some of the problems that were most frequently encountered by the author's students as they worked on extending their games. The author's school and home addresses and telephone numbers are provided. (EW)

\*\*\*\*\*  
\* Reproductions supplied by EDRS are the best that can be made \*  
\* from the original document. \*  
\*\*\*\*\*

This document has been reproduced as  
received from the person or organization  
originating it.  
Minor changes have been made to improve  
reproduction quality.

Points of view or opinions stated in this docu-  
ment do not necessarily represent official  
OERI position or policy.

## LogoWriter Games: Tips and Tools

Eadie Adamson

Copyright Eadie Adamson, 1988

Through a LogoWriter game-writing project students become involved with science, math, planning, even English (try writing clear concise instructions for a complex game!). Writing games is also an excellent approach for teaching programming skills. LogoWriter's activity cards contain a number of great starters for games, but students will inevitably want to add to their games features not included in the activity cards. More help is needed for the teacher and students if projects are not to fall apart through lack of solutions. During the past two years my students and I have found solutions for a number of game programming problems. What follows are some tips on how to get things organized and some procedures you may find helpful.

### Getting Started

Many ideas students come up with are extremely complex. Hear out the scenario, but then ask, "What's the first thing that needs to happen?" If a game can be stripped back to its simplest form, it is often not too difficult to expand it, once the initial problem is solved. Sometimes it works best to start with a single turtle, get it moving, then add another. Simplify the movement as well, even if it is to be more complicated later. Have some "generic" samples of games around as a supplement to the LogoWriter activity cards. Pull out the appropriate cards so they are easier to find.

Showing students' work from schools other than your own can provide wonderful inspiration. Psychologically, this seems a better approach than "Look what Jim did," to spur students on, when Jim is a classmate. It is easier for many students to look at and evaluate work done by people they do not know. Do share your own students' successes, but let them do the sharing (and the showing) by setting aside time occasionally to talk together about the game projects.

Encourage your students to organize themselves into teams to work together, not necessarily all on a single computer. Each team member may work on a part of the project: one for scenery, one for shapes, a few for programming. This makes it possible for even the weakest member of a group to contribute something to the enterprise.

Insist fairly early that your students learn to use CUT and PASTE to move related procedures into the order of use on the page, which will make debugging a great deal easier as the game expands. Use a superprocedure to get things started. Print out procedures from time to time as you debug.

If your students have yet to encounter the LogoWriter activity cards, try dividing your class into small groups, giving each group a different game card. Ask them to be responsible for learning how that game works and then to share their game ideas with the class. Each group could then play "teacher" for a day.

If you want to begin with the LogoWriter activity cards, the most useful for games are:

### Game Cards

Line Gobbler - Orange, No. 3  
Turtle Tag - Purple, No. 7  
Turtle Doodling - Blue, No. 1  
Hopping Turtle - Blue, No. 2

Bumper Turtles - Blue, No. 10  
Turtle Squash - Purple, No. 8  
Guess a Number - Blue, No. 1

"PERMISSION TO REPRODUCE THIS  
MATERIAL HAS BEEN GRANTED BY

Eadie Adamson

BEST COPY AVAILABLE

### Additional Resource Cards

|                                            |                                  |
|--------------------------------------------|----------------------------------|
| Drive a Truck - Dk. Green, No. 1           | Flashing Sign - Dk. Green, No. 6 |
| On the Road - Orange, No. 6                | Helicopter - Orange, No. 10      |
| Sound Effects - Red, No. 3                 | Digital Counter - Blue, No. 9    |
| Give Yourself Credit - Lt. Green, No. 10   |                                  |
| One Key Graphic Tricks - Lt. Green, No. 12 |                                  |
| Draw Thick Lines - Orange, No. 3           |                                  |

### **More Help**

Here are a few solutions for some of the problems most frequently posed by my students as they worked on extending their games. These in no way constitute a definitive collection, but should provide you with some useful "tools" for your own students.

Using the arrow keys (and other keys such as RETURN, ESC, SPACE BAR):

Type `SHOW ASCII READCHAR`, press RETURN, then press the key whose ASCII value you need. You will see a number representing the key. Invoke it in this manner (in a procedure like turtle doodling, Blue Activity Card, No. 1):

```
IF :KEY = CHAR 21 [SETH 90]
```

Or, test if a key is pressed:

```
TO GO  
FD 1  
IF KEY? [DO READCHAR]  
GO  
END
```

Write the DO procedure

```
TO DO :KEY  
IF :KEY = CHAR 21 [SETH 90]  
IF :KEY =  
    and so on...  
END
```

Remember that the DO procedure can also respond to letter and number keys; you do not need the ASCII values for these. Simply write `IF :KEY = "F [FD 10]`, and so on.

Put player instructions in the command center: `TYPE` makes text appear in the command center without a carriage return (as `INSERT` does on the LogoWriter page itself); `CHAR 13` is `RETURN`, which moves the cursor to the next line.

```
TO INSTRUCT  
CC  
TYPE [PRESS L for LEFT, R for RIGHT,]  
TYPE CHAR 13  
TYPE [F for FASTER, S for SLOWER]  
TYPE CHAR 13  
END
```

If the game uses a DO KEY procedure (above), pressing H could invoke INSTRUCT:

```
IF :KEY = "H [INSTRUCT]
```

A control key could also be programmed for help:

```
WHEN "Z [INSTRUCT]
```

### Keeping score:

Remember to initialize the score at the beginning of each game. The counter needs to go back to zero. In your INIT, or starting procedure, add a line: **NAME 0 "SHOTS**

Here is a SHOTS which counts the number of shots and ALSO prints a grammatically correct report on the number of shots ("1 SHOT" or "2 SHOTS"):

```
TO SHOTS
NAME 1 + :SHOTS "SHOTS
CT
IFELSE :SHOTS = 1 [PR SE :SHOTS "SHOT] [PR SE :SHOTS "SHOTS]
END
```

### Keeping score and stopping at maximum number:

```
TO SHOTS
NAME 1 + :SHOTS "SHOTS
END
```

At the end of the shooting procedure, add this line after the shot because the player should have a chance to make the shot and perhaps hit the target before the game stops:

```
IF :SHOTS = 3 [PR [GAME OVER] STOPALL]
```

Adding a "timer": count moves or keypresses. Here TIME counts keypresses:

```
NAME 300 "TIME

TO TIME
NAME :TIME - 1 "TIME
IF :TIME < 1
  [PR [YOU LOST] REPEAT 20 [TONE 500 55 TONE 400 55] STOPALL]
END

TO GO
IF KEY? [DO READCHAR]
FD 1
GO
END

TO DO KEY
IF :KEY = "L [LT 10 TIME]
IF :KEY = "R [RT 10 TIME]
IF :KEY = "S [SHOOT SHOTS TIME]
END
```

*Notice TIME counts every keystroke...*

Several objects need to move at once, all at different speeds. Think of a "micro-movement", the least each turtle moves at a time.

```
TO MOVEALL
TELL 0 FD :D
TELL 1 FD :D1
TELL 2 FD :D2
TELL 3 FD :D3
MOVEALL
END
```

Use an INIT procedure to set the values for the inputs to MOVEALL:

```
TO INIT
NAME 3 "D
NAME 5 "D1
NAME 1 "D2
NAME 6 "D3
END
```

Put INIT in the STARTUP or superprocedure. Call this procedure each time you play so that the initial values are established. (Scores and timers can be initialized in the same procedure.)

One way to "shoot"

```
TO PLACE
ASK 0
NAME POS "PLACE          (Stores POS belonging to turtle 0)
NAME HEADING "DIR        (Stores HEADING of turtle 0)
END

TO SHOOT
PLACE                    (Checks where turtle 0 is)
TELL 3                    (The "shot" now gets set up...)
PU SETSH 3
SETC 3
SETPOS :PLACE
SETH :DIR
FD 8 ST                  (Gets the "shot" out from under the main turtle.)
TELL 3
REPEAT 20 [FD 6 IF HIT? [STOP]]    (HIT? checks color)
HT
TELL 0                    (Control returns to Turtle 0)
END

TO HIT?
OP COLORUNDER = 2        (This outputs "True or "False to SHOOT)
END
```

The above procedures were used by some of my students. It is probably neater programming to ASK turtle 3, but they were satisfied with this.

Stopping the turtle from "wrapping" top and bottom -- and changing to appropriate shape for direction as well (this used helicopter shapes facing left and right)

|                                                                             |                                                                               |
|-----------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| <b>TO CHECK</b><br>IF YCOR > 80 [CHANGE]<br>IF YCOR < (-80) [CHANGE]<br>END | <b>TO CHANGE</b><br>IF SHAPE = 25 [SETH 90]<br>IF SHAPE = 1 [SETH 270]<br>END |
|-----------------------------------------------------------------------------|-------------------------------------------------------------------------------|

Preventing "wrapping" on a diagonal and horizontal (each turtle has a different speed):

|                                                                                                                                                 |                                                                                   |
|-------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| <b>TO MOVESTUFF</b><br>TELL 0 FD :PSPEED CKPOS0<br>TELL 1 FD :SCSPEED<br>TELL 2 FD :P2SPEED CKPOS2<br>IF KEY? [DO READCHAR]<br>MOVESTUFF<br>END | <i>(horizontal turtle)</i><br><br><i>(diagonal turtle)</i>                        |
| <b>TO CKPOS2</b><br>IF XCOR > 77 [SETH 315]<br>IF XCOR < (-77) [SETH 135]<br>END                                                                | <b>TO CKPOS0</b><br>IF XCOR > 130 [SETH 270]<br>IF XCOR < (-125) [SETH 90]<br>END |

Checking for multiple colors after a shot or during a move (and keeping score):

```

TO CHK
  ASK 1 (IF OR (COLORUNDER = 5) (COLORUNDER = 2)
    [ ...do something SCORE])
END
  
```

Setting a "level" of play, checking for empty input

```

TO LEVEL
  TYPE (Type H for HARD, E for EASY\ )
  NAME READLISTCC "LEVEL
  IF :LEVEL = [H] [NAME 50 "SPEED GO :SPEED]
  IF :LEVEL = [E] [NAME 5 "SPEED GO :SPEED]
  IF NOT MEMBER? :LEVEL [H E] [CC LEVEL]
END
  
```

Clean-up: As a last step, if the game has no graphics which must be saved, be sure it cleans up before stopping:

RG HT CT CC

Once the game is finished, if you have LogoWriter 2.0 for the Apple, fix the page the way you want it, be sure all procedures are arranged correctly, and LOCK the page! Your page will look exactly the same every time you choose it.

Eddie Adamson

School:

The Allen-Stevenson School  
 132 East 78th Street  
 New York, New York 10021  
 212 - 288-6710

Home:

1199 Park Avenue, Apt. 3A  
 New York, N.Y. 10128  
 212 - 876-3276